The Thesis Committee for Ela Pramod Joag
Certifies that this is the approved version of the following thesis:

Storage and Retrieval Technique for K-value Estimation

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor: _____
Thomas F. Edgar

Co-Supervisor: _____
Robert H. Flake

Storage and Retrieval Technique for K-value Estimation

by

Ela Pramod Joag, B.E.

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

December 2010

This work is dedicated to my parents and my husband, Ninad.

Acknowledgements

I would like to acknowledge my advisor Dr. Thomas F. Edgar for giving me an opportunity to work in his group. I am grateful to him for all his guidance and support. I would also like to thank my track advisor Dr. Robert Flake.

I am thankful to David Hill and Aaron Herrick from Chemstations Inc. for their suggestions and help in developing this work, and Chemstations Inc. in general for the financial support to this project.

I also want to acknowledge Dr. Edgar's ex-students Sidharth Abrol and John Hedengren. Their research work and knowledge were extremely helpful in this work. I am thankful to them for their prompt replies to numerous emails I sent to them.

There are few friends specially Pranav, Saurabh and my husband Ninad who have influenced this work. I would like to take this opportunity to thank them.

Special thanks to my roommates Ashwini, Jidnyasa and Shruti for making my tenure of Masters enjoyable. I specially thank my brother, Kanad for giving me motivation while taking a decision of pursuing higher studies. I am grateful to my parents, Aai and Baba, for their support in every possible way.

December 2010

Abstract

Storage and Retrieval Technique for K-value Estimation

Ela Pramod Joag, M.S.E.

The University of Texas at Austin, 2010

Supervisor: Thomas F. Edgar

Co-supervisor: Robert H. Flake

Many chemical processes need large computation time to simulate. It has been observed that the speed of a process simulation depends to a large extent on evaluation of certain important functions in the system to be simulated. One such function is K-value estimation function which involves heavy vapor-liquid equilibrium calculations. Because of large number of K-value calculations, there can be a bottleneck in simulation convergence. Thus within process simulation applications computational speed is often emphasized and accuracy is compromised.

In situ adaptive tabulation or ISAT, a storage and retrieval technique is proposed for speeding up K-value estimation in a process simulator (CHEMCAD®) using the input-output data. C language code is developed to implement ISAT algorithm for this application. The C code is then converted into Dynamic link library to be able to be used

by main CHEMCAD® code as required. The overall testing with different K-value databases gave promising results improving computational time while maintaining accuracy.

Table of Contents

List of Tables x

List of Figures xi

Chapter 1 1

Introduction 1

1.1 Reducing computation time 1

1.2 In situ adaptive tabulation 2

1.3 Overview of this dissertation 4

Chapter 2 6

ISAT Algorithm 6

2.1 Database building 7

2.2 ISAT Record Search in Binary Tree 12

2.3 ISAT Scenarios 13

2.4 Summary 20

Chapter 3 21

ISAT for K- value Estimation 21

3.1 K- value 21

3.2 ISAT for K- value estimation 22

3.3 ISAT For K-value Estimation.....23

3.4 ISAT Code as DLL.....25

3.5 Summary.....27

Chapter 4 30

Results, Conclusion and Future Work.....30

4.1 Key contributions.....30

4.2 Achievements.....31

4.3 Summary and possible future work 31

Bibliography 33

List of Tables

Table 2.1: ISAT record elements and their dimensions..... 8

Table 3.1: Specimen K-value database for 2 component system 22

List of Figures

Figure 2.1: A typical node in the tree and its two leaves 10

Figure 2.2: Addition of record to the tree 11

Figure 2.3: Addition of record to the tree 12

Figure 2.4: Ellipsoid of accuracy 15

Figure 2.5: Constant approximation EOA as a sphere with radius equal to ε_{tol} 16

Figure 2.6: Growth of an EOA 18

Figure 2.7: Constant approximation unit circle EOA 18

Figure 2.8: ISAT addition 19

Figure 3.1: Constant approximations EOA with radius equal to ε_{tol} 24

Figure 3.2: Flow of data when ISAT used as DLL 27

Figure 3.3: Initial phase of ISAT code 28

Figure 3.4: Operational phase of ISAT code 29

Chapter 1

Introduction

Most chemical processes can be mathematically modeled as a system of ordinary differential equations (ODEs), differential algebraic equations (DAEs) or partial differential equations (PDEs). Behavior of such processes can be studied by simulating the mathematical model representing the processes. Depending on the complexity of the model, these simulations may require large number of iterations to converge and in some cases may not converge at all. High computational time causes delay in calculating outputs and affects efficiency of simulation software. Thus different methodologies are being developed for computation time reduction while maintaining accuracy of the calculations.

1.1 REDUCING COMPUTATION TIME

Large-scale processes that are modeled using differential algebraic equations (DAEs) based on mass and energy balance calculations require excessive computation time to simulate. The differential equations describe material and energy balances and the algebraic equations usually describe the physical, chemical and thermodynamic properties of the system [1], [2]. One approach applied in the past is the model reduction technique. Hahn et al. [3] discussed the relevant work in nonlinear model reduction, and suggested an optimal model reduction approach which suggests capturing the dominant input-output properties of the original system at the operating point. Although the method

is simple to implement, the reduced model does not retain exact nonlinear behavior of the system. Such model reduction methods cannot reduce the number of algebraic equations. In the DAE models algebraic equations outnumber differential equations so model order reduction using such techniques is not very beneficial for reducing computational time. Hedengren et al. [1] suggested an adaptive order reduction approach for large-scale DAEs to include reduction of algebraic equations.

Any model reduction method mainly depends on the first-principles model to reduce the order, which cannot be performed using only input output data without access to the mathematical models. One more effective method to reduce the computation time is to make use of artificial neural networks (ANNs) to replace the actual model. However, one limitation of neural nets is their inability to extrapolate outside the training domain [1], [4], [5], [6], [7].

Abrol et al. [7] discussed the work that specifically deals with reducing the computational cost of actual industrial simulations, where only data can be generated by running the black-box model, without accessing the first-principles model.

1.2 IN SITU ADAPTIVE TABULATION

In many computer simulations there are several mathematical functions that use a certain number of predictive variables to determine a certain number of outputs. If the function is deterministic, the same set of predictive input variables will always produce the same set of outputs.

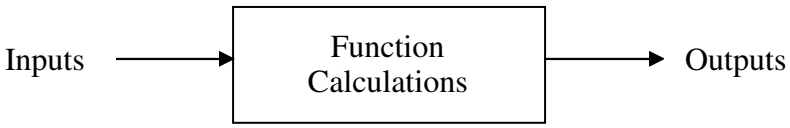

Figure 1.1 Typical function block

The block diagram in Figure 1.1 shows a typical function block representing mathematical calculations that operate on the input variables to give output variables.

In simulation models some functions are repeatedly used for different inputs. If such functions are deterministic, it is useful to store previously computed outputs of the functions for future estimation of values of outputs without doing the function calculations again and again. This method is called storage and retrieval. This method appears to be very useful when retrieval time is much less than the original function calculations.

Application of a storage and retrieval technique is In situ Adaptive Tabulation or ISAT. ISAT was originally developed for direct numerical simulation (DNS) of turbulent combustion flames. By shifting control calculations offline an efficient online retrieval is possible with significant reduction in time and computational resources. A database of input-output values is generated offline by making simulation runs in the desired range of operating conditions, and an efficient online retrieval of the stored trajectories is performed by locating a close record from the database. Thus ISAT is used for faster convergence of process simulation models. To serve the accuracy of the retrieval the estimated outputs are accepted only if they are within error tolerance (set by user) of the actual outputs. In case the estimated outputs are not within the user-defined error tolerance, ISAT automatically controls output errors, which is discussed in detail in the next chapter.

Abrol et al. [7] discussed application of ISAT to large scale dynamic simulations of a process simulator (CHEMCAD®) using the input-output data. In the work

performed the application of ISAT was extended to large-scale dynamic simulation of an actual plant scenario. Various modifications to the algorithm were suggested to improve its performance, and to adapt it to a chemical engineering problem without using the first principles model.

It is seen that the speed of a process simulation depends to a large extent on evaluation of certain important functions in the system to be simulated. If such functions are being executed repeatedly for different sets of data, then the functions can be estimated much faster by integrating ISAT algorithm to the function routines. One such function is K-value evaluation function in process simulation. K-value is Vapor liquid equilibrium ratio. The K-value calculations needed to produce a reasonable overall result for process simulation models. This dissertation discusses application of ISAT for estimation of K-values in a process simulator (CHEMCAD®) using the input-output data without using the first principles model.

.

1.3 OVERVIEW OF THIS DISSERTATION

ISAT is proposed here as a method to estimate K-values in CHEMCAD® for reducing the time required for converging complex K- value equations for different components. C language code is developed using ISAT algorithm for K-Value application. The C code is then converted into Dynamic link library to be able to be used by main CHEMCAD® code as and when required.

The ISAT algorithm is elaborated in Chapter 2. Use of binary trees for database organization and binary search method for speedy retrieval is discussed. Chapter 2 gives

key steps in the ISAT algorithm discussed in past work, e.g., how ISAT automatically controls the output error by performing the other two scenarios, *growth* or *addition,* and does not compromise on the accuracy. How this algorithm is applied to K-Value application is also discussed.

Chapter 3 introduces K values and its importance in chemical process simulation. It then extends application of ISAT to K-value estimation using input-output data only in a simulation process. ISAT is suggested as an alternate approach for faster K-value estimation without doing actual calculations, thereby improving simulation time of large-scale dynamic processes. A flow of the C code is presented to show the details. Chapter 3 further discusses use of dynamic link library and how the DLL is used for this application.

Chapter 4 discusses the results and conclusions of this thesis work. It summarizes the key contributions of the work done here, and also recommends the possible future work.

Chapter 2

ISAT Algorithm

ISAT (In situ adaptive tabulation) is a storage and retrieval method generally used to reduce the penalty of computationally expensive calculations. ISAT algorithm can be applied to approximate time-intensive computer simulations of nonlinear functions, calculations that require real-time results, or other time-intensive applications. The process involves forming a simple lookup of computed solutions (input-outputs) and efficient online retrieval from this stored lookup data. So the set of outputs can be estimated for given set of inputs without recalculating the actual function calculations.

As discussed before, ISAT is a storage and retrieval method. Hedengren et al. [8] discussed that any storage and retrieval method can be employed to a system if

- same calculations are performed repeatedly but with different values
- Retrieval is faster than original calculations
- The CPU time to generate the database is small compared with retrieval savings
- Storage cost is smaller

The algorithm essentially involves building a database of input-output values by making simulation runs offline, and then searching the database for a close record for any future queries. ISAT can perform three scenarios, namely retrieval, growth or addition, based on the closest record retrieved from the database. The three scenarios are discussed

in detail in the following sections. Depending on the desired accuracy of the trajectories the error of tolerance (ε_{tol}) on retrieval can be set by the user.

2.1 DATABASE BUILDING

Like any other algorithm that makes use of a database, the ISAT algorithm also has a phase of training associated with each application. In this training phase ISAT builds its own database using the input-output data of an application. This process of building and storing database might take time, but this is acceptable considering it is done before the online simulation runs. ISAT forms a multiple binary tree structure of the database, and once it is ready there can be faster retrievals because of binary search. The process of database building is explained in this section.

2.1.1 ISAT Record

The basic unit of the ISAT database is the record. Offline function evaluation runs generate the set of data. Using such data sets ISAT records are formed. An ISAT record is nothing but a structure of elements including a start vector, a finish vector, a sensitivity matrix, and an ellipsoid of accuracy (EOA). The record structure can be defined as

```
struct record{
   float *start;
  float *finish;
  float **sensitivity;
  float **eoa;
```

int accessed; }

Where

start is a vector of input variables;

finish is a vector of output variables

So that, finish = f (start);

sensitivity is a matrix that gives the amount that **finish** changes with a small change in **start**.

eoa is an ellipsoid of accuracy (M), which is a matrix used to approximate the estimated outputs (**finish**). EOA is elaborated in subsequent sections.

accessed is a number of times a record is used for retrieval. The table below shows the record components and their dimensions:

Record element	Dimension
start (ϕ)	$\Re^m$
finish(f)	$\Re^n$
sensitivity (A)	$\Re^{nxm}$
eoa (M)	$\Re^{mxm}$

Table 2.1: ISAT record elements and their dimensions

These records are then stored as leaves of a binary tree.

2.1.2 Binary Search Tree

The records should be organized in computer memory for efficient retrieval. If the total number of records is N, then sequential search to find a closest record will take O(N) operations to completely search the data. Instead if it is stored on binary search tree the search will take only $O(\log_2(N))$ operations. [1][7]

A binary tree is made of nodes and leaves, where each node contains a "left" pointer, a "right" pointer, and a data element. The "root" pointer points to the topmost node in the tree. The left and right pointers recursively point to smaller sub trees on either side.

The individual ISAT records are the leaves of the binary tree and a node (branch) points to two other nodes/ leaves. All branches divide until a leaf terminates the line. [6][7].

A binary search tree (BST) or ordered binary tree is a type of binary tree where the nodes are arranged in order. ISAT database is arranged in such an ordered binary tree. Each node consists of a cutting plane, defined by v and a, used to describe division between the records [7]. Figure 2.1 shows a typical node and its two leaves (record1 and record2).

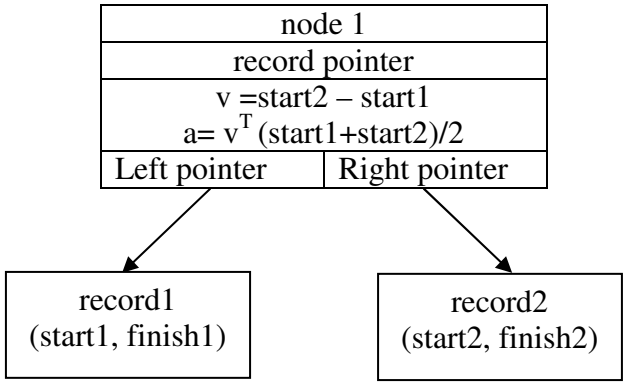

Figure 2.1: A typical node in the tree and its two leaves

A node is nothing but a structure defined as:

```
struct node {
struct record* record_ptr;
struct node* left;
struct node* right;
float *v;
float a;}
```

Where,

record_ptr is a pointer to the record to be added to the tree;

left is a pointer pointing to the left leaf of the node;

right is a pointer pointing to the right leaf of the node;

The cutting plane is defined by following two equations:

$$\mathbf{v} = \text{start2- start1} \tag{2.1}$$

$$\mathbf{a} = v^T (\text{start1+start2})/2 \tag{2.2}$$

2.1.3 ISAT Record placement in Binary Tree

An ISAT database is generated in the form of binary search tree. In this process one by one ISAT records are added to form a balanced tree. The growth of the binary tree involves the creation of a new node. Consider a case where a record to be added to the tree structure in figure 2.1 is record3. See figure 2.2. Now if **v*(start3) ≤ a** (v and a defined above) then record3 is closer to record1 and left pointer is selected. Else if **v*(start3) > a** then record3 is closer to record2 and right pointer is selected.

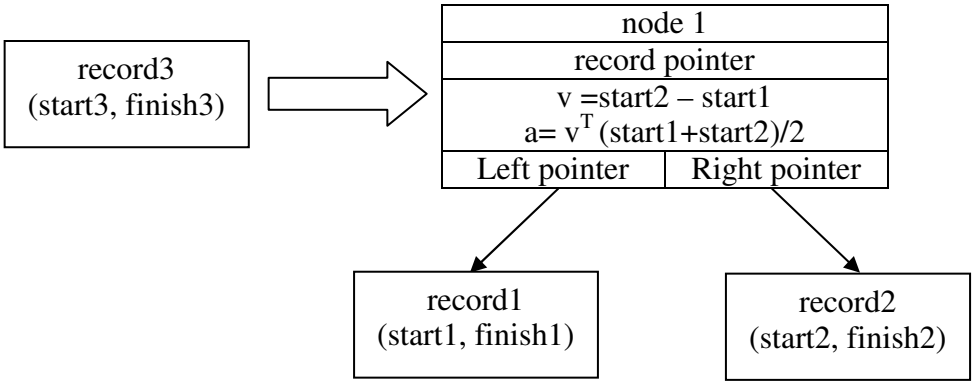

Figure 2.2: Addition of record to the tree

Supposing that record3 is closer to record2, the tree is grown on the right branch with the creation of node2. See figure 2.3.

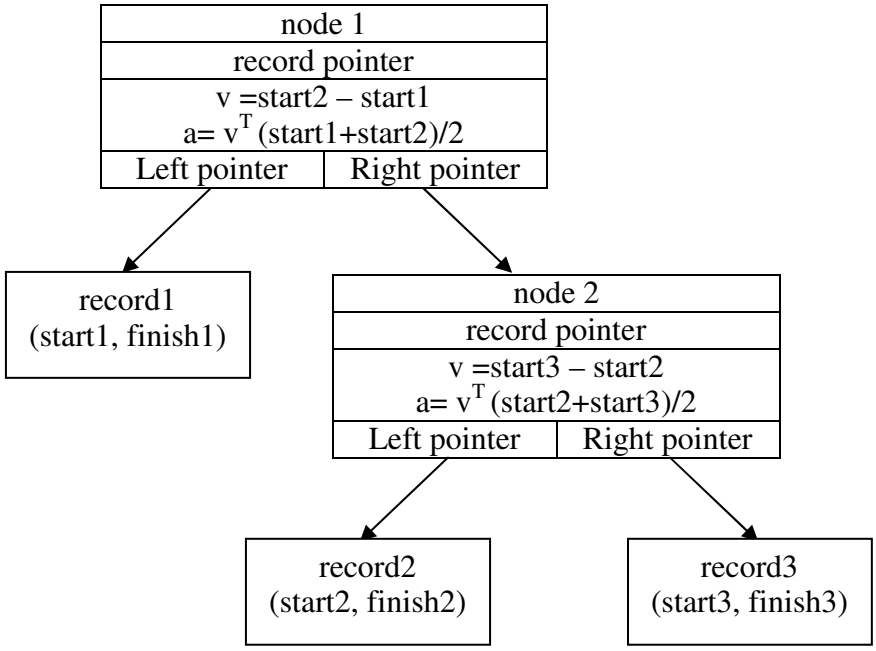

Figure 2.3: Addition of record to the tree

In this way all records are placed in the tree to prepare the ISAT database from given offline input- output data.

2.2 ISAT RECORD SEARCH IN BINARY TREE

Once the database is built, ISAT code is ready for quick retrievals of records close to the query vector. When accessing the database, the only information that is known is a query vector of predictive variables ($\mathbf{start_{query}}$). To find a close record to the query vector, the binary tree is traversed. While traversing a tree, at a node when $\mathbf{v^*(start_{query}) \leq a}$ the pointer left is selected for further search. Likewise, when $\mathbf{v^*(start_{query}) > a}$ the right pointer is selected. Searching of the database should result in retrieval of a close

record. Ideally, the closest record would be obtained by minimizing a measure of closeness. The measure of closeness is given by the difference between the query vector and the stored start vector. Thus, if the closest record searched has initial start vector as $\mathbf{start}_{\mathbf{stored}}$, the measure of closeness is

$$\mathbf{x} = |\mathbf{start}_{\mathbf{query}} - \mathbf{start}_{\mathbf{stored}}| \tag{2.3}$$

One of the drawbacks to binary tree searching is that the closest record is not always selected. To overcome this deficiency, multiple binary trees can be used to increase the probability of finding the closest record. In this case the records are equally divided among the binary trees to maintain a balance in search times. Once all of the binary trees are searched, a sequential search is performed to determine the closest record among the ones the binary trees selected. By adjusting the number of binary trees, an effective compromise is reached between the accuracy of the sequential search and the speed of the binary tree search. Abrol et al. [7] discussed more about multiple binary trees for ISAT, however for K-value application a single binary tree is efficient enough to give expected results.

Once a close record is located, ISAT performs one of three scenarios. These scenarios are retrieval, growth, and addition. These are described in more detail below.

2.3 ISAT SCENARIOS

Depending on how close a record is located while searching the database and based on a set error of tolerance (ε_{tol}), ISAT performs one of the three scenarios: retrieval, growth or addition. Before discussing about these scenarios we discuss the concept of sensitivity and EOA.

Sensitivity

As mentioned before, sensitivity matrix measures the degree to which the output changes, with small changes in inputs. Mathematically sensitivity is defined as

$$\text{Sensitivity} = \frac{\partial finish}{\partial start} \tag{2.4}$$

Once a close record has been located, the value of the function can be estimated by linear approximation using the sensitivity information. However If the sensitivity matrices cannot be estimated accurately, one can make use of 'constant approximation' instead of linearly approximating the output. In that case the sensitivity matrix $\mathbf{A} = 0$. For this application of K-value estimation the constant approximation is considered and the outputs are estimated as a finish vector of the closest record found.

Ellipsoid of accuracy

Ellipsoid of accuracy (EOA) defines the region of accuracy to control retrieval error. It is centered about $\mathbf{start}_{\mathbf{stored}}$ and consists of all $\mathbf{start}_{\mathbf{query}}$ for which the error, is less than or equal to the tolerance error, ε_{tol}. The value of ε_{tol} can be set to any value, but lower values improve the trajectory retrieval performed by ISAT. The EOA is defined by the following equation.

$$x^T M x = \varepsilon_{tol}^{\ 2} \tag{2.5}$$

where measure of closeness, $x = |\mathbf{start}_{\mathbf{query}} - \mathbf{start}_{\mathbf{stored}}|$ and M is the eoa matrix. The Figure 2.4 shows an EOA describing the region of accuracy centered on $\mathbf{start}_{\mathbf{stored}}$.

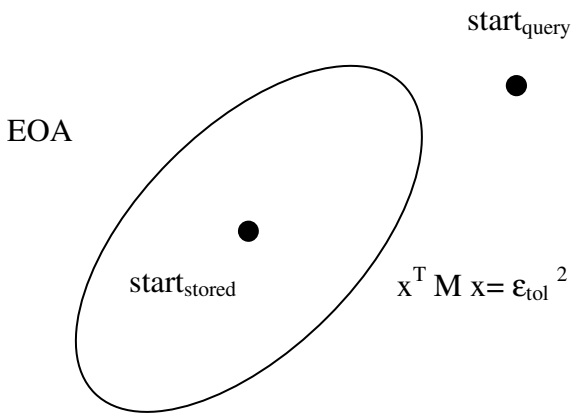

Figure 2.4: Ellipsoid of accuracy

As discussed before for K value application, a constant approximation is used by discarding sensitivity matrix (A=0). So the equivalent ellipsoid of accuracy is a circle of constant radius equal to the set error of tolerance. In such case the ellipsoids of accuracy become highly eccentric for a higher-dimensional space and for functions with high degree of nonlinearity [8]. Therefore, a better approach is to use EOAs defined by spherical surfaces with the radius being equal to the tolerance error (ε_{tol}).

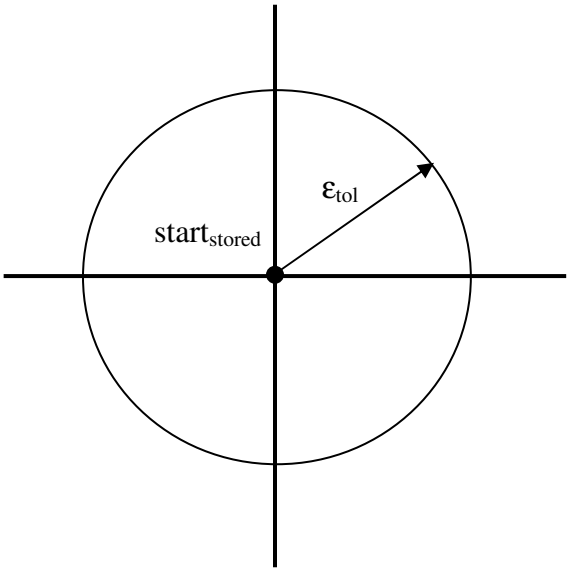

Figure 2.5: Constant approximation EOA as a sphere with radius equal to ε_{tol}

2.3.1 ISAT Retrieval

A close record to the query vector ($start_{query}$) is located by the binary search as discussed in previous section. The automatic error control decides if it is appropriate to retrieve or not. The error control is accomplished with the ellipsoid of accuracy (EOA) with a center being the $\mathbf{start_{stored}}$. The $\mathbf{start_{query}}$ point lies within the ellipsoid if $x^T M x \leq \varepsilon_{tol}^2$, where $x = |\mathbf{start_{query}} - \mathbf{start_{stored}}|$, and ε_{tol} is error of tolerance specified by user. If this is true then in case of linear approximation, finish vector is estimated by equation $finish_{est} = finish_{stored} + Ax$.

In this case of K-value estimation however by constant approximation, the stored record (with $\mathbf{finish_{stored}}$) is retrieved as it is. Thus, if the close record falls within the error of tolerance, it is retrieved as it is.

If $x^T M x > \varepsilon_{tol}^2$ then retrieval cannot be performed. It implies that $\mathbf{start_{query}}$ point is outside the EOA. So the record is not close enough and cannot be accepted. In this case the actual function calculation should take place to find out actual error.

The actual value of the function for the $\mathbf{start_{query}}$ is determined using direct integration or by running the actual simulation. Depending on the difference between the actual value ($\mathbf{finish_{actual}}$) and that estimated by ISAT ($\mathbf{finish_{est}}$) one of the other two scenarios growth or addition of ISAT is carried out.

It is important to note that there is no computational advantage with growth and addition. In both the cases a call to the actual function calculations is needed. So the actual advantage of ISAT is when the number of retrievals are far greater than growth or addition.

2.3.2 ISAT Growth

If $|finish - finish_{est}| \leq \varepsilon_{tol}$ then it means the point is outside of region of accuracy but within a specified error tolerance. So the region of accuracy (EOA) can be grown to include $\mathbf{start_{query}}$. This new region is a minimum volume ellipsoid that includes the new point, $\mathbf{start_{query}}$, and the original EOA.

As shown in figure 2.6 this growth is a minimum area expansion and includes the new point ($start_{query}$) and the original ellipse.

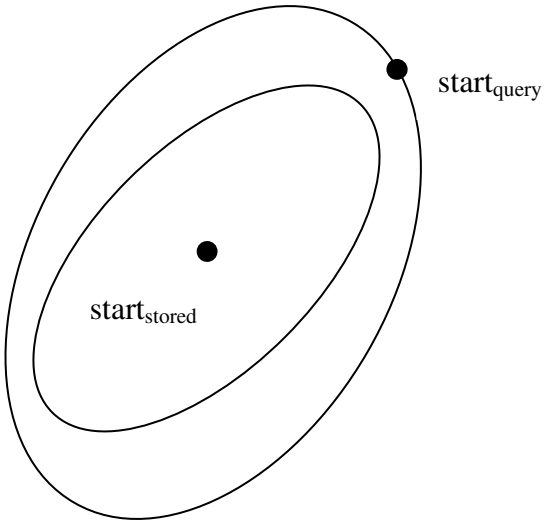

Figure 2.6: Growth of an EOA

In this case of constant approximation the unit circle can be grown to reach the query vector (as shown in figure 2.7)

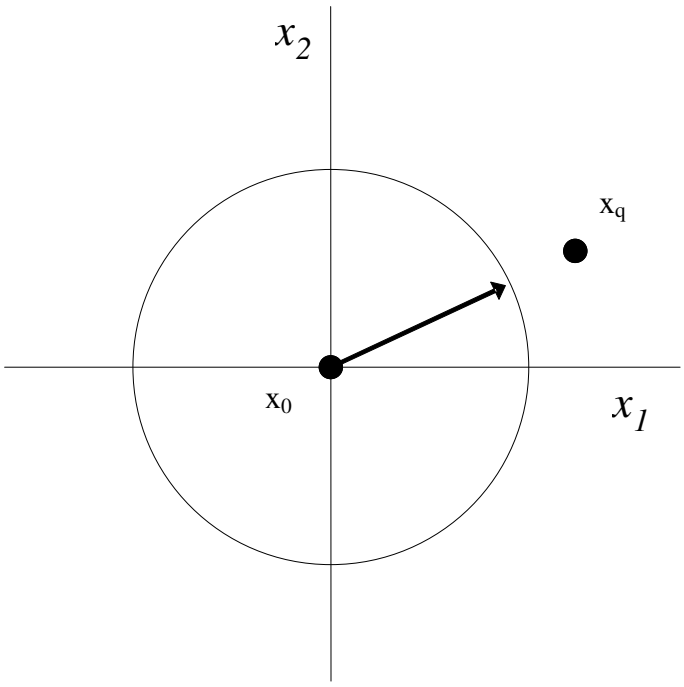

Figure 2.7: Constant approximation unit circle EOA

The growth algorithm involves alignment and growth steps. The growth point $\mathbf{start_{query}}$ becomes x_q after the translation and the EOA is defined in terms of x. Graphically, the axes are rotated so that one axis aligns with the growth point. This rotation is important so that the ellipse can be expanded along the aligned axis. When the ellipse is transformed back to the original coordinates, x_q is on the ellipse perimeter. In addition, the ellipse is a symmetric minimum area expansion that includes the growth point and the original ellipse.

2.3.3 The ISAT Addition

After actual function calculation is performed, If $|finish - finish_{est}| > \varepsilon_{tol}$ the EOA should not be expanded. Instead a new record should be added to the ISAT database. The growth step should be skipped and the algorithm jumps ahead to the ISAT addition. In the addition scenario, the record addition takes place in a same manner as discussed in section 2.1.3. This includes addition of a node with a cutting plane defined as

$$v = \phi_q - \phi \text{ and } a = v^T(\phi_q + \phi)/2$$

.

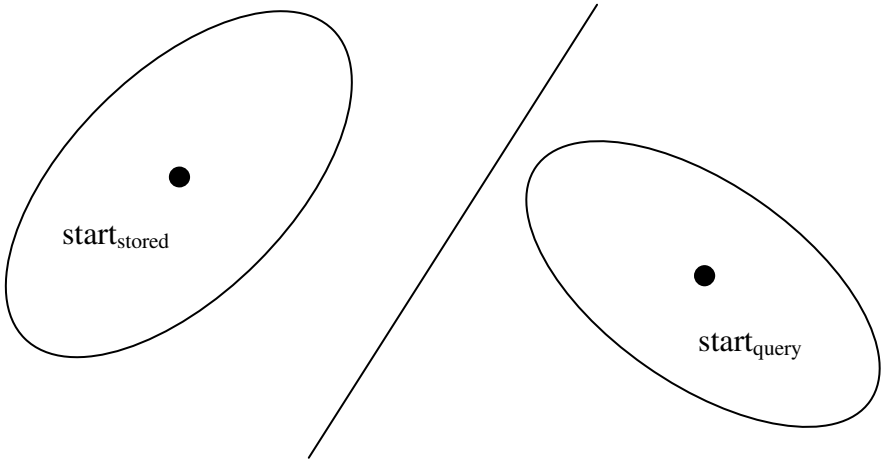

Figure 2.8: ISAT addition

2.4 SUMMARY

A detailed description of the in situ adaptive tabulation or ISAT algorithm is shown. Various aspects of the algorithm including the binary tree structure for the database and the three scenarios, retrieval, growth and addition, are discussed in detail.

Chapter 3

ISAT for K- value Estimation

The basic routine for vapor-liquid equilibrium calculations in process simulations is the temperature-pressure (T-P) flash routine. All of the well-known process simulation applications available today modify a T-P flash routine to provide results which are a tradeoff between speed and accuracy. Within these process simulation applications, computational speed is often emphasized thus accuracy is compromised.

3.1 K- VALUE

K-value is a vapor liquid equilibrium ratio. So for any given process, the K-value is a function of the vapor and liquid compositions (y_i and x_i) of all components, temperature (T) and pressure (P) (Equation 3.1). [7]

$$K = f(x_i, y_i, T, P) \tag{3.1}$$

The accuracy and speed of a process simulation depends to a large extent on evaluation of K-values for the system. Large numbers of T-P flash calculations are needed to produce converged results for process simulation models. In flash calculations, vapor-liquid equilibrium calculations in a multi-stage distillation column or calculations for systems where components undergo reactions involve frequent K-value estimations.

Because of such K-value calculations, there can be a bottleneck in simulation convergence.

3.2 ISAT FOR K- VALUE ESTIMATION

One popularly used process simulator is CHEMCAD®. ISAT is proposed here as a method to estimate K-values in CHEMCAD®. A generic database built only for the K-values of a particular system can be integrated with the calculation engine to speed-up a process simulation. A single database for the entire flow sheet with only a few variables determined by the number of components (n_c) can be built instead of considering databases for all time-intensive unit operations in the process separately. At any particular iteration, instead of calculating the K-values, an ISAT database can be used to locate a close value much faster, while maintaining the accuracy by controlling the error of tolerance.

Code was developed using C language for implementation of ISAT algorithm for K-Value estimation using only input-output data. An example of input-output data of K-value is shown in table 3.1. This is a specimen data of K-values for two components (n_c=2). So the inputs to the K-value function are temperature, pressure, x_i (mole fraction in the liquid), y_i (mole fraction in the vapor), and the outputs are K-values for each component.

Temperature	Pressure	x_1	x_2	y_1	y_2	K-value1	K-value2
709.66	29.403	0.00468	0.0004	0.00664	0.00795	0.06083	2.96241

Table 3.1: Specimen K-value database for 2 component system

With many runs on the K-value function a large database was obtained prior to the actual simulation. The C code reads the database in text format line by line and creates the ISAT database. Each reading of this text database is used to form **start** (inputs) and **finish** (outputs, here K-value) vectors of a record, where a record is an ISAT database unit discussed in Chapter 2. One by one all records are placed in the binary tree to build up the ISAT database.

3.3 ISAT FOR K-VALUE ESTIMATION

Once the ISAT database is built, the program is ready for K-value retrievals. In this case the equations for the first principles model for K-value estimation were not known so the sensitivity matrix is not considered and the outputs are estimated by constant approximation instead of linearly approximating the outputs. In that case the sensitivity matrix $\mathbf{A} = 0$, and the equivalent for the ellipsoid of accuracy is a circle of constant radius equal to the set error of tolerance. In other words the **finish** values from the retrieved record are considered as expected K-values without any linear approximation.

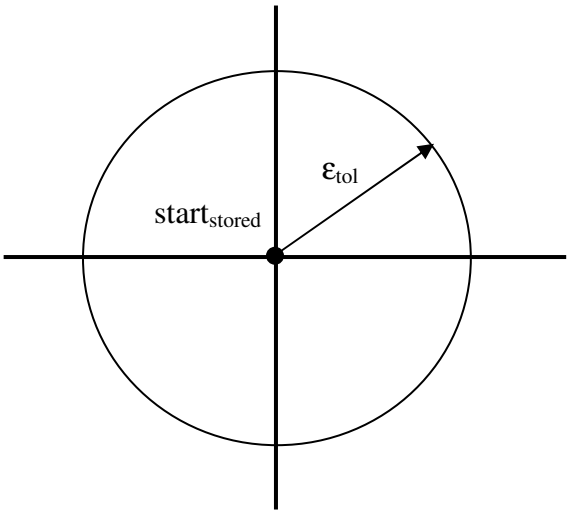

Figure 3.1: Constant approximations EOA with radius equal to ε_{tol}

3.3.1 ISAT C code Algorithm

The C program consists of four different function routines. These function routines are discussed below with their functionalities. Overall inputs to the C program are the database input file (.txt), number of components for which the database is generated (n_c), and the error of tolerance (ε_{tol}).

initialize_btree(char * input_file): This is an initialization phase of ISAT database.

This function routine reads the database text file which has the K-value database generated form of the previous runs and forms the ISAT database in a binary tree structure.

ISAT_read(float temperature, float pressure, float*fxmol, float*fymol, float fTolerance, int n_c) : Once ISAT database is ready, **ISAT_read** function is called for actual retrieval. A query vector ($start_{query}$) provided by CHEMCAD® is accepted to get

inputs (temperature, pressure, x_i, y_i). This function tries to find set of **finish** values corresponding to **start** values in the ISAT database and succeeds if finds it within ε_{tol}. If Equation 3.2 is true, then the retrieved set of data is said to be within error of tolerance.

$$(start_{query} - start_{stored})^T \; eoa \; (start_{query} - start_{stored}) \leq \varepsilon_{tol}^2 \qquad (3.2)$$

If Equation 3.2 is not true, then retrieved K-values are not within error of tolerance. In this case actual K-value function should be executed in CHEMCAD®.

ISAT_add (float temperature, float pressure, float*fxmol, float*fymol, float*fxkv, int n_c) : If retrieved K-values are not within error tolerance, then K-values are calculated in CHEMCAD®. This function creates a new record of the query vector ($start_{query}$) and calculated set of K-values, and inserts this new record into the binary tree structure. This ensures the binary tree remains updated for any future query of the same record.

ISAT_save (char * input_file): This function is called, to save the new record to the original input text file. By doing so, the record is saved permanently to the database file.

3.4 ISAT CODE AS DLL

A dynamic link library (DLL) is a collection of small programs, any of which can be called when needed by a larger program. DLL is a single file that can be shared by multiple programs at run time. A DLL file is often given a ".dll" file name suffix. DLL files are dynamically linked with the program that uses them during program execution rather than being compiled with the main program.

DLL files do not get loaded into random access memory (RAM) together with the main program, so space is saved in RAM. When a DLL file is needed, then it is loaded and run. Use of DLL saves memory and reduces swapping of files. Many processes can use a single DLL simultaneously, sharing a single copy of the DLL in memory. In contrast, for each application that is built with a static link library (SLL) Windows (or Linux) must load a copy of the library code into memory. DLL saves disk space. Many applications can share a single copy of the DLL on disk. In contrast, each application built with a static link library (SLL) has the library code linked into its executable image as a separate copy. Upgrades to the DLL are easier. When the functions in a DLL change, the applications that use them do not need to be recompiled or re-linked as long as the function arguments and return values do not change. In contrast, statically linked object code requires that the application be re-linked when the functions change.

In this application the ISAT code is used as a DLL for CHEMCAD®. Because of above advantages of DLL it is ideal to make ISAT a library file for CHEMCAD®. By doing so CHEMCAD® can use ISAT DLL as needed without integrating the code in its main code. Figure 3.2 shows the overall flow of data between CHEMCAD® and ISAT. In the ideal case CHEMCAD® sends a query vector to ISAT and ISAT finds it in its database, and if within tolerance it returns the found record to CHEMCAD®.

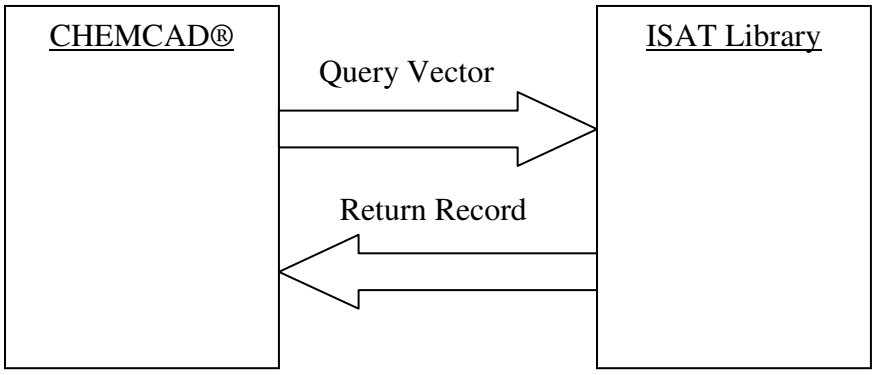

Figure 3.2: Flow of data when ISAT used as DLL

3.5 SUMMARY

K-values and its importance in chemical process simulation and the complexity of the T-P calculations are discussed in this chapter. Because of the frequency of time dependent K-value calculations there is a need to use a speedup technique at acceptable accuracy. ISAT algorithm is applied as a speedup technique to estimate K-values in CHEMCAD®. Code was developed using C language for implementation of ISAT algorithm for K-Value estimation using only input output data.

Initially, K-value database file (.txt) is used to build ISAT database (binary tree). This can be represented as shown in Figure 3.3.

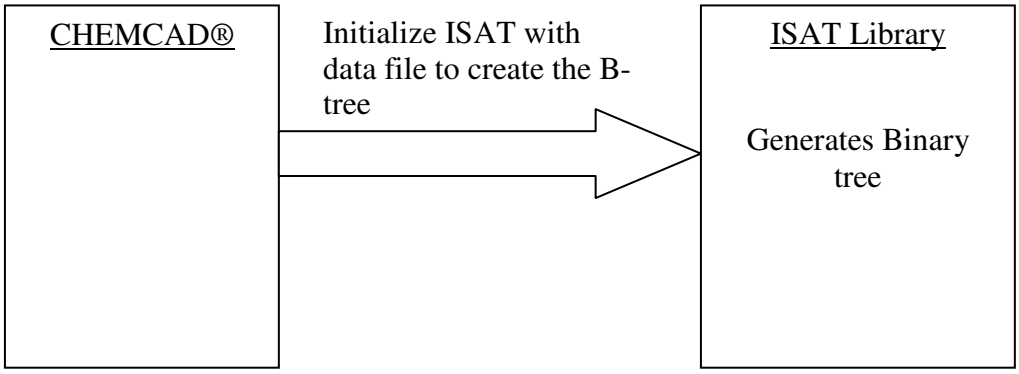

Figure 3.3: Initial phase of ISAT code

Once the database is built, the code is ready for retrievals, which is the actual operation. CHEMCAD® sends query vector, and sets error of tolerance (ε_{tol}). Accordingly a closest record is found by ISAT code and returned if within the set error tolerance. If not within ε_{tol} , CHEMCAD® can calculate K-value with rigorous calculations and call ISAT to add the calculated K-values to the database.

For the ease of operation, ISAT is used as Dynamic Link Library to CHEMCAD®, so that ISAT code can be a separate file that can be accessed as and when K-values are required. The entire process flow of the C code is shown in Figure 3.4.

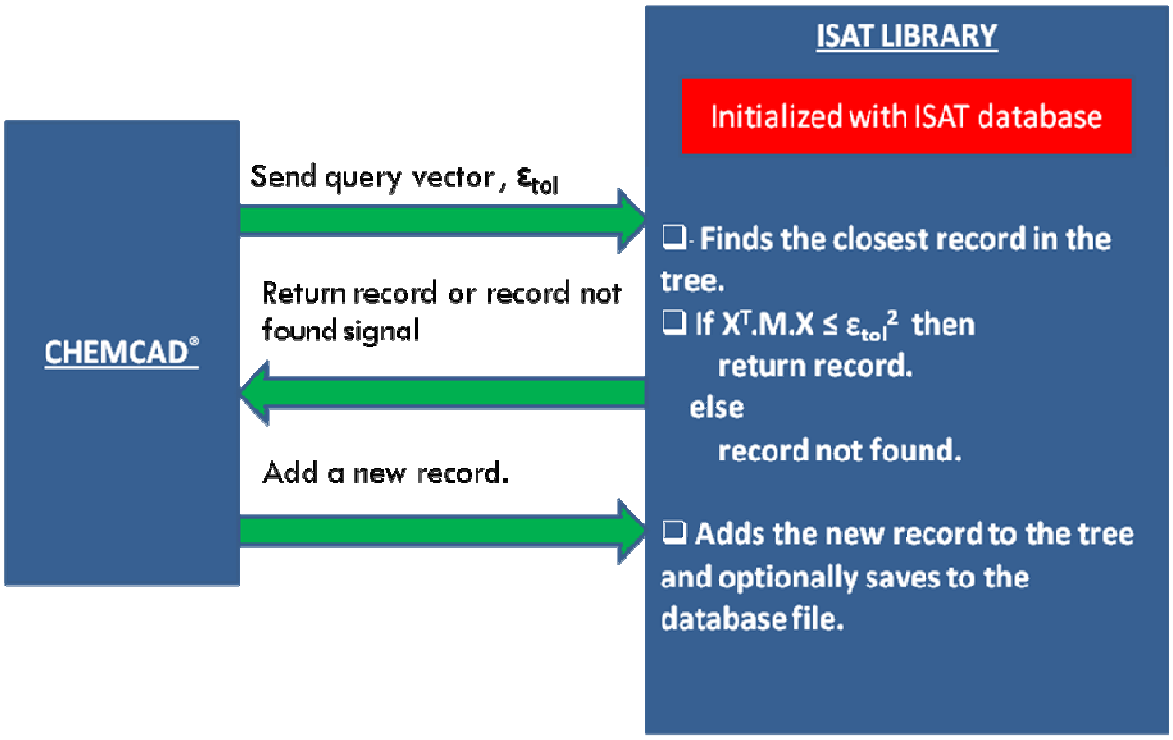

Figure 3.4: Operational phase of ISAT code

Chapter 4

Results, Conclusion and Future Work

4.1 KEY CONTRIBUTIONS

The work performed here aims at reducing the computational time required for K-value calculations in a process simulator, which is an important component of simulation of chemical processes. The algorithm used is based on the 'storage and retrieval' technique called in situ adaptive tabulation or ISAT, originally developed by Pope [9] for reducing computational load of combustion calculations.

4.1.1 C code for K-value application

Earlier the ISAT algorithm was developed in MATLAB by Abrol [7] and in FORTRAN by Hedengren [1] for different applications. The K-value calculations in CHEMCAD® take computation time as low as 1 millisecond. The real advantage of ISAT code is when the K-value estimation time is much lower than 1 millisecond. Reducing the time below 1 millisecond is difficult in MATLAB because of the high runtime. So the algorithm development for K-value application was done in C language. Some other advantages of ISAT C code are that it can be easily integrated in any code if needed and the code can be converted to library files.

4.1.2 ISAT as DLL

The ISAT C code was converted to a dynamic link library (.dll) file. A dynamic-link library (DLL) is an executable file that acts as a shared library of functions. Dynamic

linking provides a way for a process to call a function that is not part of its executable code. The executable code for the function is located in a DLL, which contains one or more functions that are compiled, linked, and stored separately from the processes that use them. DLLs save memory, reduce swapping, save disk space, upgrade easier and support multi language programs.

4.2 ACHIEVEMENTS

Application of ISAT to K-value estimation has shown promising results. The code was tested rigorously for different set of database. The ISAT code gives facility to change number of components for which K-values to be estimated. The testing was done for different number of components and for large number of readings.

The actual K-value function calculation in CHEMCAD® takes computational time of the order of 1 millisecond while implementation of ISAT gives retrieval time as low as 0.05 milliseconds. The accuracy of the retrievals is controlled by the error of tolerance (ε_{tol}), set by CHEMCAD® for each retrieval. So the computational time is reduced by 95%, giving 20 times speed up with desired accuracy.

4.3 SUMMARY AND POSSIBLE FUTURE WORK

ISAT code is proved to be an effective tool for computational time reduction. If the closest record found by the code is not within the error tolerance (ε_{tol}), then the record cannot be retrieved to maintain desired accuracy. In this case CHEMCAD® can optionally update the ISAT database by calling ISAT_add function. However the real advantage of ISAT is when the number of retrievals are more than the additions.

There can be a few possible extensions to the work done here. In this work K-value estimation is done by constant approximation, and the sensitivity matrix is considered to be 0. For more precise K-value estimation, sensitivity matrix can be calculated using a regression technique discussed by Abrol et al. [7]. When the sensitivity matrix is available K-values can be estimated by linear approximation as

$$finish_{est} = finish_{stored} + Ax \quad (4.1)$$

Where, x = where x = $|start_{query} - start_{stored}|$

The computational time would further reduced by more optimization of the existing c code.

Bibliography

[1] J.D. Hedengren and T.F. Edgar, Order Reduction of Large Scale DAE Models, *Computers and Chemical Engineering*, 29, 2069-2077, 2005

[2] J.D. Hedengren and T.F. Edgar, Approximate Nonlinear Model Predictive Control with In Situ Adaptive Tabulation, *Computers and Chemical Engineering*, 32,706-714, 2008

[3] J. Hahn, A Balancing approach to Analysis and Reduction of Nonlinear Systems, *Ph.D. Dissertation*, The University of Texas at Austin, 2002

[4] D.M. Himmelblau, Accounts of Experiences in the Application of Artificial Neural Networks in Chemical Engineering, *Ind. Eng. Chem. Res.*, 47(16), 5782-5796,2008

[5] N. Bhat and T.J. McAvoy, Use of Neural Nets for Dynamic Modeling and Control of Chemical Process Systems, *Computers and Chemical Engineering*, 14(4/5), 573-583, 1990

[6] P.S. Sastry, Memory Neuron Networks for Identification and Control of Dynamical Systems, *IEEE Transactions on Neural Networks*, 5(2), 306-319, 1994

[7] S. Abrol, Advanced Tabulation Techniques for Faster Dynamic Simulation, State Estimation and Flowsheet Optimization, *Ph.D. Dissertation*, The University of Texas at Austin, 2009

[8] J.D. Hedengren, Real-time estimation and control of large scale nonlinear DAE systems, *Ph.D. dissertation*, The University of Texas at Austin, 2005

[9] S.B. Pope, Computationally Efficient Implementation of Combustion Chemistry using In Situ Adaptive Tabulation, *Combustion Theory and Modeling*, 1, 41-63, 1997